VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2013 **Jan Hruška**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Jan Hruška**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě:CATHEDRAL Software,s.r.o.
2. Struktura závěrečné zprávy:

a. Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b. Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c. Zvolený postup řešení zadaných úkolů
d. Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e. Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f. Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeňka Chmelíková, Ph.D.**

Konzultant bakalářské práce: Ing. Josef Šustr

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

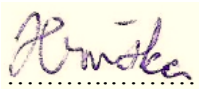
prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 3. 5. 2013



.....

podpis studenta

Poděkování

Rád bych poděkoval Ing. Zdeňce Chmelíkové, Ph.D. za vedení této bakalářské práce a Ing. Josefu Šustrovi za odbornou pomoc, konzultaci a umožnění vykonávání odborné praxe ve firmě Cathedral Software, s.r.o.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava."

Dne: *3. 5. 2013*

.....
Ing. Pavel Šustr

Abstrakt

Tato bakalářská práce popisuje průběh mé odborné praxe, kterou jsem absolvoval ve firmě Cathedral Software, s.r.o.

Obsahem práce je popis firmy a jejího zaměření včetně zadání úkolů, které jsem měl vypracovat. Hlavní část této práce se zabývá popisem řešení tvorby aplikace pro OS Windows CE zaměřenou na expedici a přeskladnění balíků.

Poslední částí práce je zhodnocení výsledků, znalosti získaných při studiu, které jsem při vykonávání této praxe uplatnil, a také těch, které jsem na praxi získal.

Klíčová slova

Odborná praxe, Cathedral Software, Basic4PPC, Windows CE, síťová komunikace, čárový kód

Abstract

This bachelor‘s thesis describes the process of my professional practice, which I passed in the company Cathedral Software Ltd.

Content of this thesis is description of the company and its profile including tasks, which I should do. The main part of this thesis deals with description of creation of an application for OS Windows CE focused on dispatch and moving packages.

Last part of this thesis is evaluation of the results, the knowledge acquired during my studies, which I applied during this professional practice and also those, which I gained.

Key words

Professional practice, Cathedral Software, Basic4PPC, Windows CE, network communication, barcode

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
IS	Information System	Informační systém
OS	Operating System	Operační systém
PC	Personal Computer	Osobní počítač
PPC	Pocket PC	Kapesní počítač

Seznam obrázků

Obsah

1 Úvod

Tato bakalářská práce popisuje absolvování individuální odborné praxe ve firmě Cathedral Software s.r.o.

Mezi vypracováním bakalářské práce na dané téma, nebo formou praxe ve firmě jsem si mohl vybrat ve 4. semestru bakalářského studia. Jelikož jsem v té době měl málo pracovních zkušeností, zaujalo mě vypracování bakalářské práce formou praxe ve firmě. Oslovil jsem několik firem nabízejících pro studenty bakalářského studia tuto formu spolupráce a v 5. semestru začal vykonávat praxi ve firmě Cathedral Software, s.r.o.

Tato firma nabízela práci na různých úkolech, ze kterých mě zaujalo programování aplikace pro mobilní terminál s integrovaným snímačem čárových kódů a OS Windows CE 5.0. Tento úkol pro mne byl zajímavý zejména z toho důvodu, že mne vždy zajímalo programování pro mobilní zařízení a již na střední škole jsem se podílel na projektu aplikace pro OS Windows Mobile, který je na Windows CE založený.

Aplikaci, kterou jsem měl za úkol naprogramovat, lze rozdělit do 2 hlavních funkcí – expedici a přeskladnění balíků. Před zahájením programování samotné aplikace jsem ve spolupráci s panem Ing. Josefem Šustrem odzkoušel různé možnosti tvorby aplikací pro tento mobilní OS. Z nich byl zvolen programovací jazyk a stejnojmenné vývojové prostředí Basic4PPC, které vyvíjí firma Anywhere Software.

Tato práce se dále dělí na 4 hlavní části – informace o firmě a mém pracovním zařazení, podrobné zadání úkolů, postup jejich řešení a zhodnocení dosažených výsledků.

2 Popis firmy Cathedral Software, s.r.o.

2.1 Historie

Společnost Cathedral Software, s.r.o. vznikla v polovině roku 1996 jako sdružení fyzických osob Cathedral Software & Hardware. Sídlo společnosti je již od založení ve městě Prostějov. Firma se od počátku zaměřuje na dlouhodobou spolupráci, tvorbu softwaru na zakázku a poskytovaní ostatních služeb dle požadavků zákazníků.

V souvislosti s požadavky zákazníků firma rozšířila portfolio poskytovaných služeb o dodávku počítačových sestav a komponent. V letech 1997-1998 firma začala s dodávkami, instalacemi a správou informačních systémů a dále s návrhem, realizací a výstavbou počítačových sítí a realizací virtuálních privátních sítí. V roce 1999 firma přesídlila do prostor v obchodním centru Atrium, rozrostla se o nové grafiky a programátory a začala poskytovat služby v oblasti webdesignu.

Především po roce 2000 se firma intenzivněji zaměřovala na vývoj informačních systémů tvořených podle přání a potřeb zákazníků. Jedním z prvních byl IS Žaluzie a Hokej 2000. V roce 2004 firma získala zakázku na informační systém pro výchovné ústavy, který je v současnosti úspěšně implementován.

Zlomovým informačním systémem byl pro firmu IS ArisCAT, který firma vyvíjí již od roku 2001. Systém je specifický především optimalizováním na zpracování obtížně konfigurovatelných výrobků.

Mezi úspěchy ve vývoji informačních systémů patří také zahájení spolupráce se společností LCS International a.s., která patří mezi největší producenty podnikových aplikací na českém trhu. Cílem spolupráce je vývoj a distribuce modulu pro IS Helios zajišťujícího zpracování obchodů s těžko konfigurovatelnými výrobky, jako jsou např. žaluzie, rolety, vrata, plastová okna.

V roce 2007 byla firma přeměněna na novou právní formu a tímto vznikla společnost Cathedral Software s.r.o.

2.2 Pracovní zařazení

Ve firmě jsem pracoval na pozici programátora aplikace pro mobilní terminál s integrovaným snímačem čárových kódů. Mým úkolem bylo vytvořit aplikaci, která bude sloužit pro expedici a přeskladnění balíků. Vytvořená aplikace by měla po síti komunikovat s IS ArisCAT.

3 Zadání úkolů v průběhu praxe

Zadáním praxe bylo naprogramování aplikace pro mobilní terminál s integrovaným snímačem čárových kódů s OS Windows CE. Aplikace by měla mít 2 funkce - expedici a přeskladnění balíků.

Společná pro obě funkce je komunikace s IS ArisCAT prostřednictvím protokolu TCP/IP a přenosu textových příkazů. Zadáním bylo vytvořit aplikaci, kterou je možné ukončit jen po načtení nebo zadání kódu a zůstane spuštěna i po uspání zařízení. Aplikace měla mít možnost zadání a uložení IP adresy a portu serveru do souboru a po jejím opětovném spuštění údaje načíst. Po spuštění aplikace by měla být zobrazena hlavní obrazovka, na které je možno zvolit jednu z funkcí, případně změnit uložené IP adresy a porty daného serveru. Po zvolení některé funkce se aplikace pokusí připojit k příslušnému serveru a v případě, že je server nedostupný, nebo zařízení není připojeno k bezdrátové síti, v daném časovém intervalu se pokus o připojení opakuje. Stejným způsobem se aplikace zachová i v případě přerušení spojení.

Pro testování aplikace mi firma zapůjčila mobilní terminál s integrovaným snímačem čárových kódů ARGOX PT-6020.

3.1 Expedice balíků

Funkce expedice balíků by měla sloužit pro označení balíků jako naložených v IS ArisCAT. Aplikaci a způsob její komunikace se serverem bylo třeba přizpůsobit funkčnosti stávajících čteček, které neobsahují standardní mobilní OS a budou využívány společně s novými. Z toho důvodu byla vyloučena jakákoliv úprava funkčnosti na serveru. Ze stejného důvodu bylo, namísto přímé komunikace s databází, využito přenosu jen jednoduchých textových příkazů a informací.

Po spuštění aplikace a připojení k serveru aplikace čeká na informace o zakázkách, které jsou v IS ArisCAT označeny k expedici. Kvůli omezení ze strany stávajících čteček server zasílá údaje o maximálně 8 zakázkách. Údaje o každé zakázce zahrnují číslo zakázky, celkový počet balíků a počet balíků, které ještě zbývá naložit. Informace o zakázkách jsou poté zobrazeny na displeji, a pokud již jsou naloženy všechny balíky některé zakázky, server ji dále nezasílá a v následující zprávě do seznamu doplní jinou zakázku s nenaloženými balíky.

Po načtení informací o zakázkách aplikace čeká na načtení čárového kódu, který poté zasílá serveru. Podle odpovědi serveru si aplikace vyžádá nové informace o zakázkách a přehraje tón oznamující úspěšné označení balíku, případně informuje obsluhu, že již byl naložen, nebo není určen k expedici a přehraje varovný tón. O nové informace o zakázkách aplikace žádá také periodicky, aby např. v případě naložení balíků jiným zařízením měla obsluha k dispozici aktuální informace.

3.2 Přeskladnění balíků

Funkce přeskladnění balíků je určena pro přemístění balíků podle zakázky tak, aby po přemístění byly balíky určité zakázky pohromadě na jednom či více stojanech a IS ArisCAT měl informace o jejich aktuálním umístění. Způsob komunikace se serverem zůstává stejný jako u funkce expedice balíků. Rozdílem oproti první funkci je, že veškeré texty, které budou ve formulářích této funkce zobrazeny, podléhají jazykovému překladu a aplikace je bude načítat ze serveru.

Po spuštění aplikace a výběru funkce přeskladnění se aplikace připojí k serveru a očekává od něj zprávu s textem pro přihlašovací formulář. V tomto formuláři se následně může obsluhující pracovník přihlásit zadáním nebo načtením svého kódu. Po úspěšném přihlášení dojde k zobrazení formuláře pro načtení čísla zakázky. V tomto formuláři bude kromě samotného načtení čísla zakázky umožněno také odhlášení pracovníka.

Po načtení platného čísla zakázky bude zobrazen hlavní formulář této funkce, který obsahuje informace o aktuálním umístění balíků dané zakázky a obsluhuje jejich přemístění. Při přemístění určitého balíku dochází k načtení čísla zdrojového stojanu, ze kterého je balík přesouván a cílového stojanu, na kterém bude balík umístěn po přesunutí. Tento přesun je možné s dalšími balíky zakázky opakovat a čísla stojanů, na kterých jsou balíky aktuálně umístěny, odeslat až po přemístění více nebo všech balíků. Ve formuláři je možnost čísla zdrojových i cílových stojanů smazat, a tím přesun balíků zrušit, případně celý formulář zavřít a vrátit se na formulář pro načtení čísla zakázky.

4 Použité nástroje

4.1 Možnosti vývoje aplikací pro Windows CE

Po seznámení se se zadáním úkolů bylo třeba vybrat směr, kterým se při vývoji aplikace ubírat. I když OS Windows CE podporuje spouštění souborů s příponou „exe“, není na něm možné jednoduše spustit aplikaci určenou pro desktopový OS.

Z důvodu pozvolného ústupu tohoto OS z mobilních zařízení v posledních letech jsem po konzultaci hledal řešení, které by bylo multiplatformní. Po ověření, že firmou využívané vývojové prostředí sice podporuje vývoj aplikací pro Android, iOS i Windows Phone, ale už ne pro Windows CE, mne napadlo realizovat zadání jako webovou aplikaci. Vytvořil jsem jednoduchý formulář a zobrazil jej na čtečce v Internet Exploreru. Problém tohoto řešení spočíval právě v integrovaném prohlížeči, jehož grafické rozhraní zabírá velkou část malého displeje. Alternativní internetové prohlížeče se nainstalovat a zprovoznit nepodařilo a od tohoto řešení bylo po konzultaci upuštěno.

Další možností bylo naprogramování aplikace v jazyku C++ nebo C# s využitím .NET Frameworku. Tato možnost nakonec, z důvodu nemožnosti vývoje ve Visual Studiu 2008 Express a využití .NET Frameworku, zůstala nevyužita.

Firmou navrhnutým řešením bylo použití software Aplikační generátor od společnosti Codeware, která jej dodává k zakoupeným zařízením. Jak už název napovídá, vytvoření aplikace je v něm sice velmi jednoduché, ale možnosti tvorby jsou omezené. Software nebylo možné použít např. z toho důvodu, že pro jakoukoliv komunikaci je nutné využít SQL server.

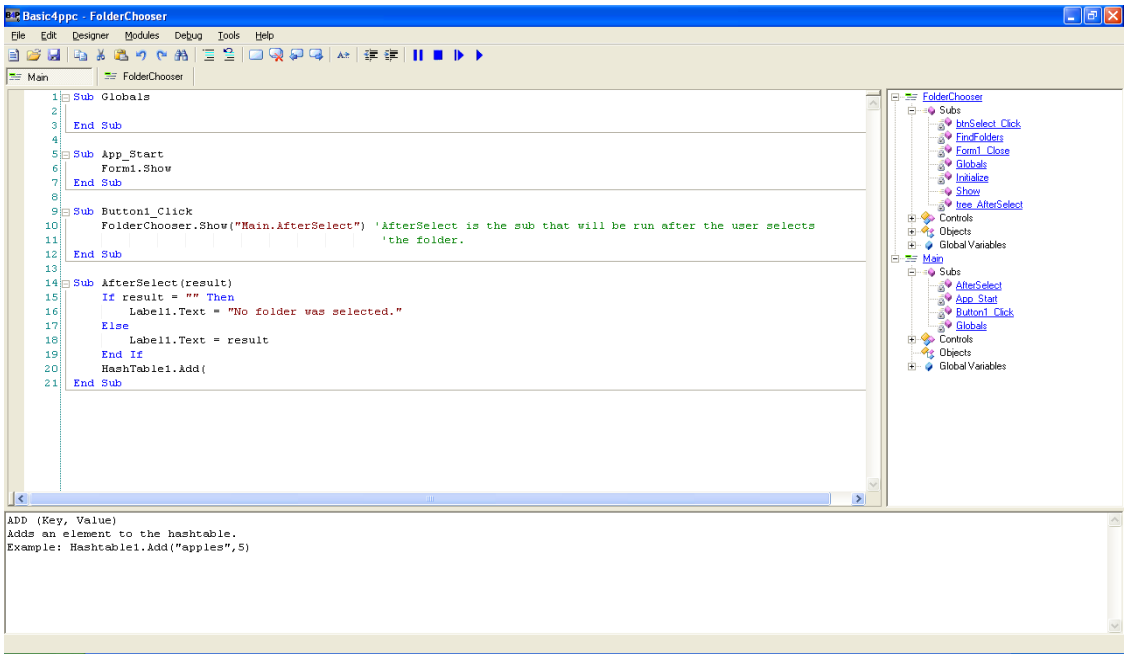
Nakonec byl pro tvorbu aplikace zvolen programovací jazyk a stejnojmenné vývojové prostředí Basic4PPC, které popisuje následující kapitola.

4.2 Basic4PPC

Basic4PPC je programovací jazyk vyvíjený od roku 2005 společností Anywhere Software. Poslední verze byla vydána v roce 2010 a společnost se od této doby zaměřuje především na produkt Basic4android.

Pro spuštění programů napsaných v Basic4PPC je vyžadován nainstalovaný Microsoft .NET Framework na PC a .NET Compact Framework na zařízení s Windows CE/Mobile. Basic4PPC podporuje dotyková i nedotyková zařízení, ale prostředí pro mobilní zařízení vyžaduje dotykovou obrazovku. Výhodou programování v Basic4PPC je, že vytvořené aplikace jsou spustitelné i na PC s Windows bez použití emulátoru, s výjimkou těch, které používají knihovny určené pouze pro mobilní zařízení. Některé knihovny jsou univerzální, ostatní mají různé verze pro použití na Windows a Windows CE/Mobile. Většina knihoven nutných pro vytvoření aplikací různého zaměření je obsažena již v základu, a pokud některá funkcionalita standardním knihovnám schází, je možné využít některou z mnoha knihoven, které vyvíjí komunita na oficiálním fóru. Basic4PPC je možné před zakoupením zdarma vyzkoušet, ale v této verzi není možné používat komunitou vytvořené knihovny, ani vytvořenou aplikaci zkompilovat pro mobilní zařízení. Přídavné knihovny je možno vytvářet v C#, nebo ve Visual Basicu. Basic4PPC používá v základu kódování UTF-8 a je pro něj k dispozici

podrobná dokumentace jak v samotném vývojovém prostředí, tak na internetových stránkách společnosti.



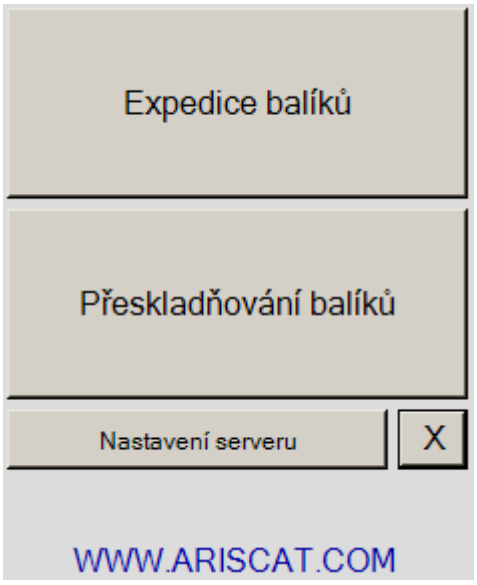
Obrázek 4 .1: Prostředí Basic4PPC

5 Postup při řešení úkolů

Po seznámení se se zadáním a výběrem nástrojů pro vývoj aplikace jsem se nejdříve zaměřil na samotné spojení aplikace se serverem, které je pro obě funkce společné.

Jelikož jsem se s Basic4PPC setkal již dříve, věděl jsem, že na internetových stránkách tohoto produktu je k dispozici jak podrobná nápověda ke knihovnám dostupných již po instalaci, tak ukázky některých vytvořených aplikací. Zaměřil jsem se na knihovnu Network a s pomocí ukázkových kódů poměrně jednoduše zprovoznil aplikaci, kdy klientská i serverová část je napsána v Basic4PPC a umožňuje jednoduchý přenos zpráv po síti. Tato aplikace využívala komponenty Client a Server zajišťující samotné navázání spojení a knihovnu BinaryFile, jež se používá také např. pro čtení a zápis do souboru. Práce s touto knihovnou spočívá nejprve ve vytvoření spojení (stream) např. k souboru nebo k navázanému síťovému spojení. Tato knihovna obsahuje funkce pro odesílání i příjem dat různých datových typů. Využitím objektu této komponenty je přiřazení navázaného síťového spojení funkcí `Client.GetStream`.

Poté jsem vyzkoušel připojení k testovací serverové aplikaci napodobující chování firemního IS ArisCAT. Navázání spojení bylo sice úspěšné, ale odesílání a příjem dat již ne. V nápovědě pro knihovnu Network s využitím BinaryFile byly pro příjem a odesílání textových dat použity funkce `ReadString` a `WriteString`. Při použití funkce `WriteString` pro odeslání dat, ale server nepřijal data korektně a při příjmu dat s použitím funkce `ReadString` program skončil chybou. V nápovědě knihovny jsem našel funkce `WriteBytes` a `ReadBytes`, které umožňují odesílání a příjem dat ve formě pole bytů. Pro převod textového řetězce na pole bytů a naopak poté slouží funkce `StringToBytes` a `BytesToString`. S použitím těchto funkcí bylo odesílání i příjem dat funkční a bylo potřeba jen upravit kódování a přidat zasílání znaku klávesy Enter, který na konci každého řetězce očekává server.



Obrázek 5 .1: Formulář aplikace zobrazený po spuštění

Před prací na jednotlivých částech jsem do aplikace přidal formulář a funkce pro načtení a uložení IP adres a portů serverů z a do souboru. S pomocí nápovědy ke komponentě BinaryFile bylo zprovoznění jednoduché a funkční prakticky napoprvé. Navíc bylo potřeba přidat jen ověření existence souboru při spuštění aplikace funkcí `FileExist()`.

5.1 Expedice balíků

Když bylo sestavení spojení a jednoduchá komunikace se serverem funkční, zaměřil jsem se na část aplikace určené pro expedici balíků.

Pro zobrazení informací o načtených zakázkách, kterých server zasílá maximálně 8, jsem zvolil prvek Table, jehož rozvržení je definováno až ve zdrojovém programu. Informace o každé zakázce sestávají z čísla zakázky, celkového počtu balíků a počtu balíku, které ještě zbývá naložit. Při připojení aplikace k serveru se proto vytvoří tabulka o 3 sloupcích a zašle na server požadavek na informace o zakázkách. Dále je v aplikaci Timer, který dvakrát za vteřinu kontroluje, zda klient nepřijal nová data. Použití Timeru je nutné, protože samotná komponenta Client nemá implementovánu událost pro příjem dat. Identifikovat přijetí dat je možné pouze ověřením stavu vlastnosti `Client.DataAvailable`. Po příjmu odpovědi se zpracované informace o jednotlivých zakázkách naplní do tabulky. Aktualizace tabulky probíhá nejprve jejím vyprázdněním a poté postupným plněním po řádcích funkcí `AddRow`.

Stěžejní funkcí aplikace pro expedici balíků je načítání jejich čárových kódů a následně zasílání na server. Zapůjčený mobilní terminál již obsahoval aplikaci, která komunikuje se snímačem čárových kódů a umožňovala nastavit převedení načteného kódu na stisky hardwarových tlačítek. To znamená, že z hlediska jakékoliv aplikace je poté načtení čárového kódu stejné jako jeho zadání na klávesnici. Za načtený kód ještě daná aplikace, která je spuštěna vždy na pozadí, vkládala znak klávesy Enter. Této funkce jsem využil původně použitím komponenty HardKeys, která umožňuje zachycení stisku některých hardwarových kláves v aplikaci. Po prozkoumání možností této komponenty jsem bohužel narazil na její limit, jelikož umožňuje zachycení stisku jen kurzorových kláves, Enteru a numerických kláves 1-5. Jediné řešení pro zachycení stisku i zbytku numerických kláves, které jsem nalezl, by omezilo funkčnost aplikace jen na zapůjčené zařízení, a tak jsem zkusil na formulář vložit TextBox, který má, mimo jiné, implementovánu událost `KeyPress(Key)`, přičemž pomocí funkce `Asc(Key)` je možné získat ASCII kód stisknuté klávesy. Vyplnění načteného kódu bylo v režii obslužné aplikace, a tak tato událost rozeznává jen stisk klávesy Enter, která kód ukončuje. Po jejím stisknutí je načtený čárový kód odeslán na server. Kód události jsem následně upravil tak, aby pole pro načtení čárového kódu nebylo vyprázdněno ihned po jeho odeslání, ale až před načtením jiného kódu. Tato úprava umožňuje případnou kontrolu správnosti načteného čárového kódu.

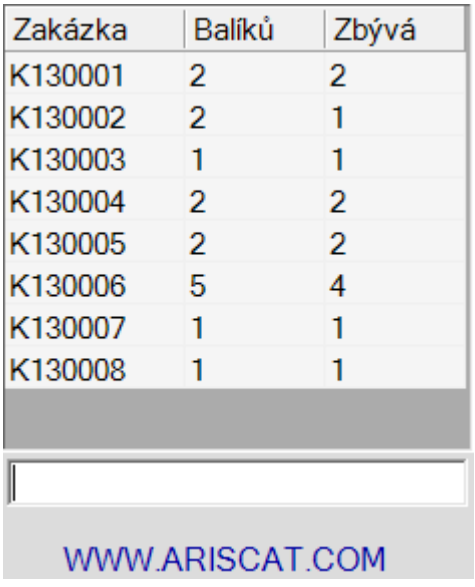
Při testování aplikace bylo nutné nejprve kliknout na dané textové pole a teprve poté načíst čárový kód. V aplikaci jsem tedy určil focus na textové pole ihned po zobrazení formuláře a opět jej obnovil v události `LostFocus`. Díky tomu je focus aplikace držen stále na textovém poli.

Po zajištění funkčního zobrazení informací o zakázkách a odeslání načteného kódu balíku jsem se zaměřil na zachycení odpovědi od serveru jako reakce na přijatý kód balíku. V případě, že byl

kód platný, aplikace reaguje přehráním oznamujícího tónu a odesláním požadavku na aktuální informace o zakázkách. V opačném případě zasílá server ve zprávě i text chybové hlášky, takže aplikace reaguje přehráním varovného tónu a zobrazením chybové zprávy s textem dané hlášky. Zobrazení chybových zpráv ovšem bylo nutné pozdržet o délku varovného tónu, protože při zobrazení jakékoliv chybové zprávy systém přehraje krátký tón, nezávisle na aplikaci, která vyvolala zobrazení dané hlášky. Při pokusné změně pořadí těchto úkonů bylo docíleno pouze přehrání tónu až po zavření zobrazené hlášky.

Pro případ, že by např. byly některé balíky naloženy jiným zařízením, byl do aplikace přidán Timer, který v intervalu 10 vteřin odesílá na server požadavek na zaslání aktuálních informací o zakázkách. Následně, kvůli možnému nastání situace, kdy aplikace odešle dva požadavky v krátkém intervalu za sebou (po naložení balíku a v rámci Timeru), jsem do aplikace přidal kontrolu odeslaných požadavků a zpracovaných odpovědí. Pokud tedy aplikace na server odešle požadavek a ještě nebyla zpracována odpověď, je odeslání dalších požadavků blokováno.

Aplikace měla podle zadání běžet v režimu celé obrazovky, ale po jeho nastavení a spuštění aplikace na čtečce se v horní části obrazovky zobrazil šedý pruh překrývající i část tabulky s informacemi o zakázkách. Podle informací z fóra [3] je tento šedý pruh vyhrazen pro zobrazení menu aplikace a to i v případě, že žádné menu nebylo vytvořeno. Opravou byla změna argumentů funkce pro nastavení daného formuláře do režimu celé obrazovky. Protože při běhu aplikace v režimu celé obrazovky není zobrazena horní lišta, upravil jsem událost textového pole pro načtení kódu tak, aby se při načtení určeného kódu aplikace ukončila. Tímto je také zamezeno vypnutí aplikace obsluhujícím pracovníkem. Na následujícím obrázku je zobrazen konečný vzhled formuláře pro expedici balíků.



Obrázek 5 .2: Formulář pro expedici balíků

Jednou z časově nejnáročnějších částí bylo přizpůsobení chování aplikace tomu, že bude obsluhujícím pracovníkem neukončitelná. To znamená, že po spuštění aplikace a výběru funkce zůstane neustále spuštěna a musí být připravena jak na výpadek spojení, tak uspání zařízení nebo

nedostupnost serveru. Případnou nedostupnost serveru nebo bezdrátového připojení jsem řešil přidáním Timeru, který se spustí po prvním neúspěšném pokusu o připojení. V intervalu 10 vteřin se poté aplikace pokouší o nové navázání spojení. Protože tento proces trvá relativně dlouhou dobu cca 7 vteřin, po které aplikace nereaguje, a je zbytečné jej provádět v případě nedostupnosti bezdrátového připojení, vytvořil jsem funkci pro rozpoznání, zda je zařízení připojeno k síti. Toho je docíleno pomocí vytvoření testovacího serveru jen pro potřeby klienta, který zjišťuje jeho aktuální IP adresu. Pokud při pokusu o opakované připojení k serveru není dostupná bezdrátová síť, a tedy má testovací server IP adresu localhostu, nezdržuje se aplikace pokusem o spojení a zobrazí informaci o nedostupnosti síťového připojení. Stejná situace nastane, pokud dojde k výpadku již navázaného spojení, navíc jsou zastaveny ostatní Timery a po obnovení spojení se aplikace vrátí do stavu před výpadkem.

Po otestování aplikace s ostrým serverem byla zjištěna chyba týkající se uspání a následného probuzení zařízení. Chyba se zobrazila jen v některých případech a příčinou je samotný režim spánku zařízení. Namísto vypnutí zařízení, které OS Windows CE žádným standardním způsobem neumožňuje, je umožněno pouze uspání zařízení. Režim spánku OS Windows CE funguje podobně jako na standardním PC s OS Windows, a tak v případě, že před přechodem zařízení do režimu spánku byla spuštěna některá aplikace, je po opětovném spuštění zařízení běh aplikace obnoven. To se ale netýká síťových připojení, která jsou v režimu spánku neaktivní. Ve spojitosti s tím, že komponenta Client nerozpozná výpadek spojení, se po spuštění zařízení nepravidelně objevovala chyba v kódu Timeru, který zajišťuje zpracování dat přijatých od serveru. Protože je obvykle spojení po spuštění zařízení opět rychle obnoveno, přidal jsem do kódu Timeru toleranci tří cyklů, ve kterých není možné ověřit dostupnost přijatých dat. Pokud chyba trvá více než tři cykly, je Timer zastaven a aplikace se chová stejně jako při výpadku spojení.

Poslední nedostatek aplikace byl zjištěn až při jejím nasazení v ostrém provozu a primárně se týkal nefunkčnosti obnovování informací o zakázkách v intervalu 10 vteřin. Tento problém se mi nedařilo nasimulovat a objevil jsem jej až po zaslání mnoha požadavků v krátké době za sebou, kdy se v chybové hlášce označující, že daný balík již byl naložen, zobrazila i další zpráva s informacemi o zakázkách. Situace tedy nastala jen v případě, pokud bylo aplikací odesláno číslo balíku a hned poté byla provedena metoda Timeru, ve které se na server zasílá požadavek na zaslání informací o zakázkách. Server sice v takovém případě přijal dva požadavky, na které odeslal dvě odpovědi, ale kvůli intervalu Timeru, který kontroluje přijatá data jen dvakrát za vteřinu, byly odpovědi spojeny. Na tuto situaci nebyla funkce pro zpracování přijatých dat připravena a druhou odpověď nezpracovala. Protože aplikace obsahuje kontrolu odeslaných požadavků a přijatých odpovědí, dokud např. na požadavek na informace o zakázkách není přijata odpověď, je odeslání dalších požadavků blokováno. Problém nastal při spojení dvou odpovědí, kdy druhá z nich obsahovala informace o zakázkách, a tedy nebyla zpracována a kontrolní funkcí označena jako přijata. Jedním z možných řešení bylo snížení rizika vzniku této situace snížením intervalu Timeru, který kontroluje přijatá data. Interval Timeru by ale kvůli výkonu zařízení, nebylo možné výrazně snížit, a tak jsem se zaměřil na funkci pro zpracování přijatých dat a identifikaci spojených odpovědí. Pro jejich identifikaci jsem využil toho, že na konec každé zaslané odpovědi server přidává znak klávesy Enter. Při přijetí odpovědi od serveru nejprve funkce ověří, zda je tento znak na konci textového řetězce, nebo uvnitř a v tom případě podle něj řetězec rozdělí. Následně jsou po rozdělení zpracovány jednotlivé odpovědi.

5.2 Přeskladnění balíků

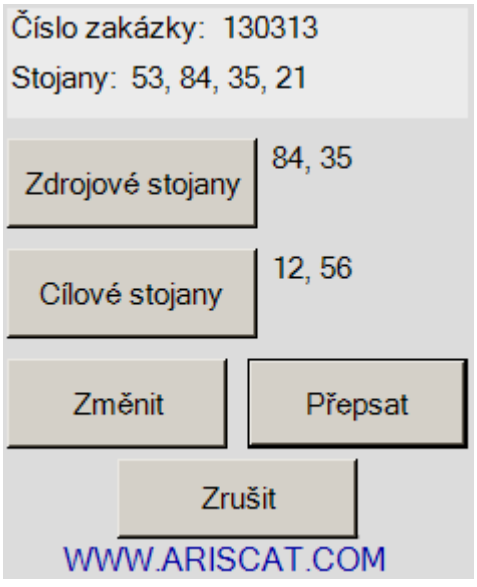
Funkce pro expedici a přeskladnění balíků mají podstatnou část týkající se komunikace se serverem a řešení problémů s tím spojených společnou. Proto jsem se při řešení této části mohl zaměřit jen na funkci přeskladnění a i přes její rozšířenější možnosti byla tato část časově méně náročná.

Po vytvoření počátečních verzí jednoduchých formulářů jsem upravil funkci pro zpracování přijatých odpovědí od serveru tak, aby uměla zpracovat odpovědi specifické pro tuto funkci. V těchto odpovědích jsou zasílány i texty podléhající jazykovému překladu, které jsou poté použity pro popis textových polí a tlačítek ve formulářích.

Po stisknutí tlačítka pro přeskladnění balíků v hlavním formuláři se aplikace pokouší o navázání spojení se serverem. Pokud by se připojení k serveru nezdařilo, stejně jako u funkce expedice balíků se aplikace dále pokouší o připojení v intervalu 10 vteřin. Po připojení k serveru aplikace zašle inicializační zprávu. Server na ni odpoví s textem pro přihlašovací formulář, který aplikace teprve po zpracování odpovědi zobrazí.

Obsahem přihlašovacího formuláře je, kromě jeho názvu, jen pole pro vyplnění nebo načtení kódu obsluhujícího pracovníka. Ten aplikace odesílá k ověření serveru, a pokud je platný, v odpovědi opět zasílá texty pro formulář určený k načtení čísla zakázky, který je poté zobrazen. V případě, že by zaslaný kód byl neplatný, aplikace přehraje varovný tón a zobrazí chybovou zprávu přijatou v odpovědi od serveru. Řešení problému se zastavením přehrávání varovného tónu po zobrazení chybové zprávy je obsaženo v předchozí kapitole. Následující formulář obsahuje pole pro načtení čísla zakázky a tlačítko pro odhlášení, po jehož stisknutí aplikace přejde zpět do přihlašovacího formuláře. Načtené číslo zakázky aplikace opět odesílá serveru, který odpoví, zda je nebo není platné. Při odpovědi od serveru označující odeslané číslo zakázky jako neplatné, dochází stejně jako v případě chyby při přihlášení, k přehrání varovného tónu a zobrazení přijaté chybové zprávy.

V odpovědi po načtení platného čísla zakázky server zasílá čísla stojanů, na kterých jsou balíky zakázky aktuálně umístěny a texty v příslušném jazyce pro vytvoření formuláře určeného pro přeskladnění balíků. Vzhled tohoto formuláře znázorňuje následující obrázek.



Obrázek 5 .3: Formulář pro přeskladnění balíků

V horní části tohoto formuláře je zobrazeno číslo zakázky a čísla stojanů, na kterých jsou aktuálně umístěny její balíky. Po stisknutí tlačítka Zdrojové stojany je zobrazen jednoduchý formulář, ve kterém je možné načíst číslo stojanu, ze kterého je balík odebrán. Po načtení čísla stojanu se toto číslo uloží do paměti, aplikace se přepne zpět do formuláře pro přeskladnění a doplní do něj číslo načteného stojanu. Po stisknutí tlačítka Cílové stojany je zobrazen stejný jednoduchý formulář, ve kterém je možné načíst číslo cílového stojanu, na který je balík přemístěn. Poté je číslo načteného cílového stojanu opět uloženo do paměti a doplněno do formuláře pro přeskladnění, do nějž se aplikace vrátí. Načítat čísla zdrojových a cílových stojanů je možno opakovaně, přičemž po načtení čísla každého stojanu aplikace kontroluje jeho duplicitu.

Tlačítko Změnit slouží pro odeslání seznamu načtených zdrojových a cílových stojanů na server. Server poté odpovídá, zda bylo přeskladnění úspěšně zapsáno, případně chybovou zprávu o neplatnosti čísla zakázky nebo některého stojanu. V případě chyby je přehrán varovný tón a zobrazena přijatá chybová zpráva. Pokud server přeskladnění úspěšně zapíše, je s jeho potvrzením odeslán i seznam aktualizovaných čísel stojanů. Tento seznam je poté zapsán do textového pole v horní části formuláře a smazány seznamy zdrojových a cílových stojanů.

Tlačítko Přepsat slouží pro odeslání pouze seznamu cílových stojanů. V tom případě je celý seznam stojanů, na kterých jsou balíky umístěny, zrušen a nahrazen odeslaným seznamem cílových stojanů. Server opět ověřuje platnost čísla zakázky a stojanů, a pokud je nový seznam úspěšně zapsán, server jej pošle v následující odpovědi zpět.

Tlačítko Zrušit má dvě funkce a popisky. Pokud již ve formuláři bylo načteno číslo některého stojanu, slouží pro smazání seznamů načtených zdrojových i cílových stojanů. V případě, že jsou oba seznamy prázdné, má tlačítko popisek Zavřít a po jeho stisknutí se aplikace vrátí do formuláře pro načtení čísla zakázky.

Po dopracování funkce pro přeskladnění balíků jsem vytvořil testovací server, na kterém jsem testoval případné chyby. Většina chyb byla vyřešena ve funkci pro expedici balíků, a proto bylo třeba dopracovat jen chování aplikace při výpadku spojení a po jeho obnovení. Pokud k výpadku spojení dojde před přihlášením nebo načtením čísla zakázky, je až do jeho obnovení načítání kódů a čísla zakázky blokováno. Pokud k výpadku spojení dojde při zobrazeném formuláři pro přeskladnění, je do obnovení blokováno stisknutí tlačítek Změnit a Přepsat. Po obnovení spojení aplikace navíc zašle na server požadavek na zaslání aktuálního seznamu stojanů.

6 Znalosti a dovednosti, které jsem uplatnil

Po celou dobu této odborné praxe jsem programoval ve vývojovém prostředí Basic4PPC, se kterým jsem se v rámci jiného projektu setkal již na střední škole.

Při vytváření zadané aplikace jsem uplatnil především znalosti týkající se analýzy dané aplikace, algoritmů a programovacích technik obecně a znalost programování pro mobilní zařízení. Dále jsem uplatnil znalost psaní přehledného zdrojového kódu, který později mohou upravovat programátoři firmy. Nakonec jsem se snažil vytvořit jednoduchý vzhled aplikace aplikace s ohledem na zásady pro tvorbu uživatelských rozhraní.

7 Znalosti a dovednosti, které mi chyběly

Při programování zadané aplikace jsem často využíval nápovědu použitého vývojového prostředí Basic4PPC a oficiální fórum. To se týká zejména úvodní fáze, ve které jsem řešil připojení a způsob jednoduché komunikace se serverem.

Před započetím programování bylo také nutné ověřit možnosti zachycení načteného čárového kódu, ať už přímou komunikací s použitým snímačem či použitím obslužné aplikace. Pro určení správného postupu řešení některých problémů jsem také musel částečně zjistit, jak funguje samotný OS Windows CE, obzvláště chování aplikací a síťových připojení po přechodu zařízení do úsporných režimů.

8 Závěr

Možnost absolvování odborné praxe místo psaní bakalářské práce jsem si vybral proto, že jsem si chtěl vyzkoušet práci na pozici programátora v praxi.

Během absolvování této praxe jsem získal zkušenosti spolupráce, která byla důležitá jak při konzultacích o zadání a postupech řešení, tak pro následné odhalení a odstranění některých problémů. Získal jsem také přehled o dění v softwarové firmě a důležitosti další spolupráce se zákazníkem.

Z odborných znalostí jsem se především naučil programovat pro mobilní zařízení se systémem Windows CE a tyto znalosti mohu později využit částečně také při programování pro zařízení se systémem Android, pro který je možno vytvářet aplikace ve vývojovém prostředí Basic4android.

Výsledkem této odborné praxe je vytvoření aplikace určené pro expedici a přeskladnění balíků pro použití především ve firmách, které se zabývají výrobou oken, žaluzií, rolet, vrat apod. Funkce pro expedici balíků je již nasazena a úspěšně používána v praxi a k využití funkce pro přeskladnění balíků v praxi dojde po termínu odevzdání této bakalářské práce. S firmou proto spolupracuji nadále a budu řešit případné nastalé problémy s aplikací.

Použitá literatura

[1] *Cathedral Software* [online]. 2013 [cit. 2013-05-05]. Dostupné z: http://www.cathedral.cz/

[2] *Basic4ppc - Windows Mobile programming and Pocket PC Development* [online]. 2013 [cit. 2013-05-05]. Dostupné z: http://www.basic4ppc.com/specifications.html

[3] *Basic4ppc (Windows Mobile) - Development Forum* [online]. 2013 [cit. 2013-05-05]. Dostupné z: http://www.basic4ppc.com/forum/basic4ppc-windows-mobile/